INTERACTIVE TIMESHAPING OF MUSICAL SCORES WITH BACH AND CAGE

Daniele Ghisi
University of California, Berkeley
CNMAT
danieleghisi@berkeley.edu

Andrea Agostini
Conservatorio di Torino
andreaagostini@conservatoriotorino.eu

Eric Maestri
Conservatorio di Genova
eric.maestri@conspaganini.it

ABSTRACT

In this article we aim to organize the collection of practices that amount to modifying the temporality of a musical score in order to create a new one, by representing them as a combination of three basic families of operations: dilations, distorsions and repetitions. The initial score can be borrowed or designed on purpose, and can be of any complexity (an entire score, a single note, or anything in between). We call the complex of these practices 'timeshaping'. We describe the analytical space derived from the proposed organization and show how many of its processes can be implemented in an interactive computer-aided composition environment using the *bach* and *cage* libraries for Max.

1. INTRODUCTION

In this paper we focus on a set of practices that we call 'timeshaping'. These practices are defined by the transformation of a pre-existing musical representation—such as a score—through the reorganization in time of its elements.

More specifically, timeshaping can be thought of as a 'symbolic processing technique' and, as such, it can be applied to both specifically conceived musical materials and pre-existing ones. In a way, similarly to the signal processing techniques, these two possibilities mirror the way an audio effect, such as a filter, can be employed both as a part of a synthesizer, thus being instrumental to the production from scratch of a new sound object, and as a step of the processing chain applied to a prerecorded sound sample. If pushed to the extreme, these processing techniques, in both the audio and the symbolic domains, can become essential tools for working within the broad conceptual area of musical borrowing [1, 2], which defines an operative approach laying between the traditional compositional strategies of reinterpretation of pre-existing material and the manipulation of a corpus of data that can be performed in real time. One may refer to these two compositional attitudes, which define two conceptual poles with a vast area of overlapping between them rather than a clear, black-and-white opposition, respectively *'tabula rasa'* and *'tabula plena'* approaches [3].

Timeshaping, potentially as an instance of an ampler, yet-to-be-defined family of symbolic processing techniques, shares some other traits with sound processing techniques: for example, it can be both fluid and written; it can conceptually be performed in both real and deferred time; its parameters can be defined both through experimentation and formalisation; and it is an abstract operation (or, rather, a family thereof) which is cognitively significant, and substantially independent from the details of its practical implementation. Each of the techniques described in the following sections can indeed be expressed through a variety of tools, insofar as these tools provide ways of representing and manipulating symbolic musical events: these include all the major software systems for computer-aided and algorithmic composition, such as OpenMusic [1] [5], PWGL [6] and Abjad [7]. However, we shall discuss one specific implementation of timeshaping practices, the only comprehensive one we are aware of, that is, a subset of the *cage* package for Max [8], developed by two of the authors as a complement and an expansion of the *bach* package.

1.1 *bach* and *cage*

bach [9] is a set of modules for Max (both external objects written in C/C++ and abstractions developed in Max itself) establishing a few new data types aimed at representing symbolic musical data, a large set of tools for operating upon those data types, and some objects for displaying, editing graphically and playing back the musical scores they represent. *cage* [10], on the other hand, is a Max package built upon *bach* and taking advantage of its features. It implements a set of musically meaningful operations ranging from middle-level utilities (such as the parsing of SDIF files) to high-level musical processes (such as generation of scales and arpeggios, or symbolic ring modulation).

One of the main, overarching principles of the *bach* ecosystem is real-time reactiveness: generally speaking, *bach* complies with the real-time data-flow computational paradigm of Max, which, although quite complex from a theoretical point of view, fosters a novel kind of relation with computer-aided composition, one in which the computer becomes a real musical instrument [11, 12].

[1] Our approach is fundamentally different from the one proposed in [4], where the author, under the term 'timesculpt', describes a series of analytical and combinatorial techniques applied to rhythms. In particular, our approach does not separate the rhythmic component from the other musical parameters, but rather treats each score as a whole.

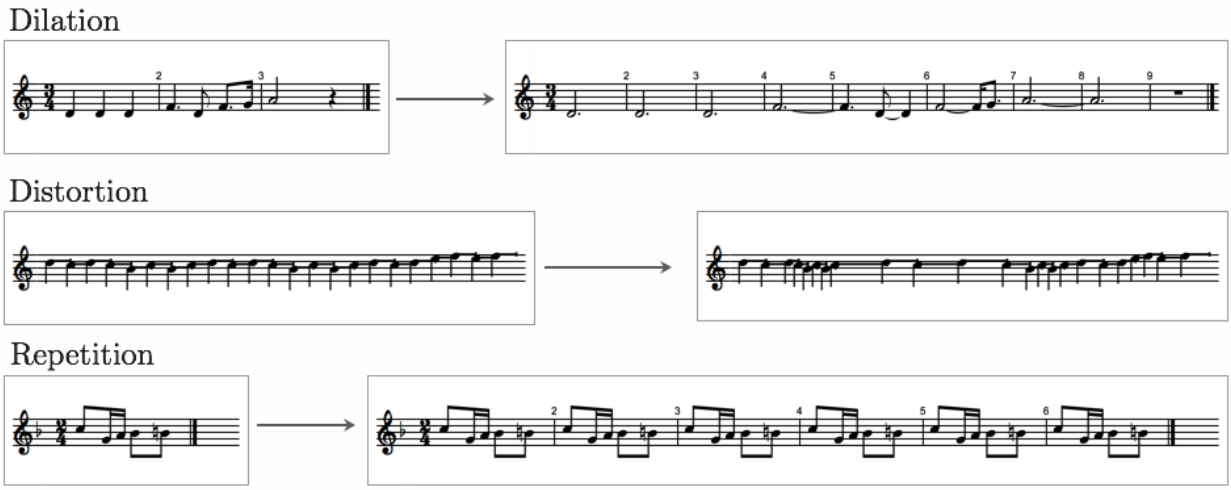

Figure 1. A simple example for each of the main timeshaping modifications. From top to bottom: augmentation from a section of Josquin's *Missa L'homme armé (Agnus Dei)*; temporal distorsion of the beginning of Bach's Cantata BWV 249a; repetition of a figure in Beethoven's 6th Symphony (1st movement, measure 17, first violins).

1.2 Relationship with compositional time

As Kramer suggests, time has a dual nature [13, p. 18]. On the one hand, it is linear, when it concerns becoming; on the other hand, it is non-linear, since it is thought in its spatial dimension. Linear time defines the flow that goes from the past to the future passing through the present; nonlinear time is visualised, represented by musical notation. Composers usually deal with both types of time: non-linear time (the composed one) is a model for linear time (the perceived one).

Libraries such as *bach* and *cage* allow a specific type of interaction between these compositional times. The computer becomes a real musical instrument for sketching and experimenting with direct feedback in the hands of the composer. As a consequence, the musical representation status changes and can be considered as a performed musical object. Conceiving notation as a 'temporal instrument' [14], *bach* and *cage* allow to reduce the distance between linear and non-linear time, thanks to 'symbolic-sonorous objects' that can potentially be manipulated performatively. These libraries offer an interactive 'graphematic space' [15, p. 215] which is, at the same time, a representational space.

2. TIMESHAPING

We call 'timeshaping' a collection of compositional practices that modify the temporal shape of a musical score (borrowed or designed on purpose), or a part of it—an entire score, a single note, or anything in between—in order to create a new one. These timeshaping practices are common when applied to audio buffers, but less so when applied to symbolic musical representations. We can distinguish at least three orthogonal axes, representing different kinds of modifications (Fig. 1):

1. **Dilation**: uniformly stretch or shrink the content of the source score by a given factor. Sometimes the factor is inferred in order to match a given target duration. A classical example of usage of pure dilations are augmentation or diminution canons. If the factor is negative, the dilation is also combined with a retrogradation (reversal).

2. **Temporal distortion**: warp the content of the source score internally, without changing its overall duration. The operation is non-linear and may happen in a variety of different ways. It is accomplished by reading the source score at a variable rate, under the condition that the overall elapsed time stays unchanged. Rhythmic interpolations between different musical figures often fall in this category. As a side note, we can consider rhythmic quantization as a very specific type of distortion, approximating a proportionally notated score with a traditionally notated one.

3. **Repetition**: replicate the content of the source score at regular intervals in time.

One may attempt to use the space determined by the three orthogonal axis as a map on which to organize a number of timeshaping processes: such an attempt yields, in our view, Fig. 2. Such a diagram does not aim to any mathematical consistency, rather it is meant to be an informal analytical tool. For one thing, the dilation axis represents both the amount of 'stretching' and the amount of 'shrinking'. Secondly, temporal distortions can of course be so complex that representing them on a single axis makes for a drastic approximation; it should also be remarked that the constraint by which a distortion may not cause a modification of the overall duration of the original score can be seen as an arbitrary one, but it is useful to clearly distinguish it from dilation; indeed, most typical agogical processes, such as accelerandos and rallentandos, involve both categories. Finally, when repetitions are mixed with other modifications, there may be ambiguity on whether the modification is to be applied to all instances equally or can be applied

with different parameters to different instances—we shall abide by the latter interpretation.

It should be clear to this point that the axes of this space do not constitute a representation of the actual parameters of one specific process, but they will hopefully provide a conceptual framework for discussing such processes in terms of few elementary operations.

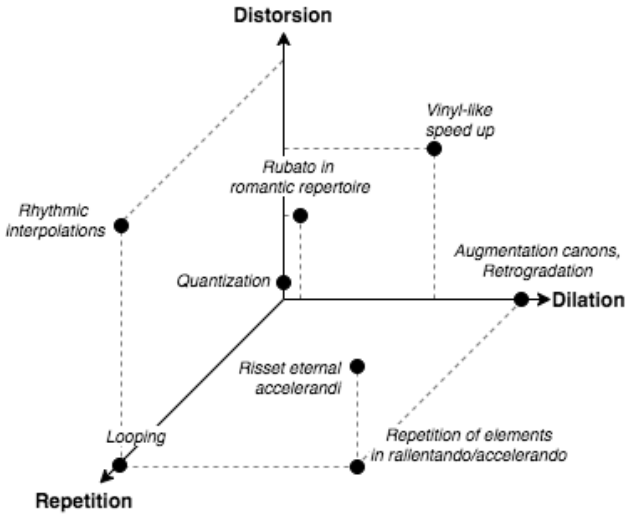

Figure 2. A diagram showing some timeshaping practices organized by amount of dilation, repetition and distortion.

3. TIMESHAPING PRACTICES IN CAGE

A portion of the *cage* library has been designed to allow users to manipulate the temporality of scores. We will describe their general underlying ideas and how they relate to the representation proposed in the previous section; their scope of application with respect to the three main axes is represented in Fig. 6. For all that follows, let t_L be the time of the source score (sometimes called 'logical time'), and let t_E be the elapsed time in the output score.

3.1 Dilation

The simplest module allowing the modification of the temporality of a source score is *cage.timestretch*: it stretches the incoming score either by a fixed factor or in order to match a given target duration (Fig. 3), i.e., $t_E = k \cdot t_L$. The module works both on proportionally notated scores, as represented by the object *bach.roll*, and on traditionally notated ones, as represented by *bach.score*. In the latter case, the dilation factor has to be a rational number [2], and the user has the ability to choose how meters in the output score should be handled.

3.2 Temporal warping

The *cage.timewarp* module performs more refined local modifications of the temporality of a source score; in its general form, it is able to reassign notes and markers to some other position in time, usually defined through the usage of a 'transfer function' f, in a lambda loop configuration [16]. [3] In short, the object module from its rightmost 'lambda' outlet the original position in time t_L of every item (such as a note or a marker) and expects to receive in its rightmost 'lambda' inlet either its desired resulting temporal position or the rate of the remapping. More specifically, the module operates in two in two different ways according to the value of the 'order' attribute:

- With 'order 0', it performs a time-to-time mapping: for each original temporal position, given in milliseconds, the transfer function returns a new temporal position at which the corresponding musical items must be remapped, as in $t_E = f(t_L)$. For instance, if the function f is such that $f(1000) = 2000$, then a chord with onset at 1 second will have its onset at 2 seconds in the resulting score. If $f(t_L) = kt_L$ is a linear function of slope k, then the module coincides with *cage.timestretch*.
- With 'order 1', it maps every source time (on the x axis) to the *rate* of the remapping, i.e., to the derivative $r(t_L) = dt_L/dt_E$. For instance a function $f \equiv 1$ will simply yield the same source score; a function $f \equiv k$ will result in a timestretch of a factor $1/k$; linear functions yield uniform accelerandi/rallentandi, and so on.

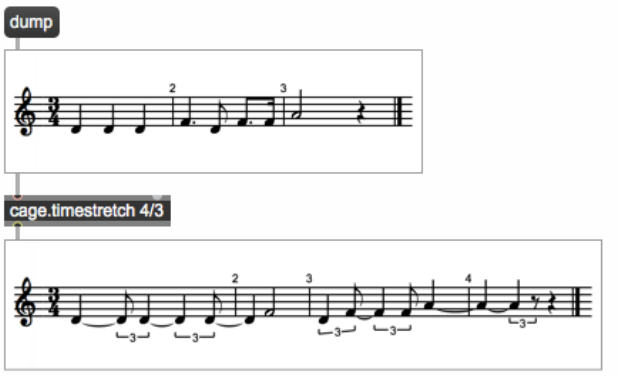

Figure 3. A simple example of dilation of a musical score obtained via *cage.timestretch*.

It is interesting to remark that *cage.timewarp* with 'order 0' provides a quite direct implementation of the dilation and distortion axes of the timeshaping conceptual space described above, in that distortions correspond to non-linear functions of time against time, whereas dilations generally correspond to linear functions. Assuming $f(0) = 0$, no

[2] Rational numbers are not implemented in Max per se, but they are one of the data types added by the *bach* package.

[3] A lambda loop is an idiomatic patching configuration typical of several *bach* and *cage* objects and abstractions: one or more dedicated outlets of a module output data iteratively to a section of the patch that must calculate a result (such as a modification of the original data or a boolean return value) and return it to a dedicated inlet of the starting object [16]. Lambda loops are an attempt at providing a way to implement custom specialization of generic processes, in a similar way to what happens with anonymous function in several programming languages, in the strictly non-functional paradigm of Max. For example, sorting a list of MIDI pitch numbers by pitch class can be achieved through the *bach.sort* object with a lambda loop containing a modulo 12 operation.

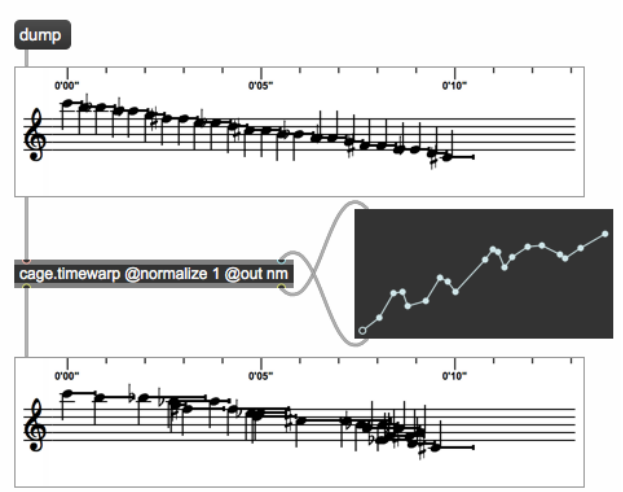

Figure 4. An example of temporal distortion of a descending chromatic scale obtained via *cage.timewarp*. Notice how the order of notes can also become shuffled.

dilation takes place if and only if $f(D_L) = D_L$ with D_L being the duration of the source score (if the object is set to normalized mode, this means that $f(1) = 1$, thus including for instance all the functions $f(x) = x^\alpha$). The linear function $f(x) = x$ means that no temporal modification has been performed, and corresponds to the origin of the timeshaping space.

The module can be also used in a 'normalized' mode, so that the source time t_L is received in the normalized range $[0., 1.]$, 0 corresponding to the beginning of the score and 1 to its end. This normalized usage makes it easy to perform pure temporal distortion, without modification of the overall score duration (see Fig. 4).

In a typical usage scenario, the lambda loop of *cage.timewarp* contains a graphical breakpoint function, rather than an algebraically defined one. Especially in this case, the 'order 1' case can be seen as more intuitively musical, as higher values on the y axis correspond to faster tempi, rather than the somehow more abstract notion of more advanced positions in the score.

With this in mind, this module can be seen as a flexible tool for representing various shapes of accelerando or rallentando, the latter of which can be pushed to the extreme case of producing a 'negative' tempi, that is, retrograde readings of portions of a score. Consistently with the overall design of *bach* and *cage*, which both put a strong focus on the reactivity of operations, an interesting aspect of *cage.timewarp* lies in that it allows to perform a real-time experimentation with the temporality of the original score, one in which the composer can interactively modify the transfer function, either graphically or through some parameter, and immediately receive from the machine a feedback showing the result. It is our position that this kind of interactivity represents in itself a novel paradigm in the field of computer-aided composition.

3.3 Varied repetition

The *cage.repeat* module produces identical repetitions of a source score (either in proportional or in standard notation). The *cage.looper* module goes further, providing the possibility to only loop a portion of score (and not the entire one), and to allow for each iteration to be individually modified, according to a lambda loop mechanism: each loop iteration (in *bach.roll* format) as well as its duration are output through the lambda outlets of the module: the user can provide a custom mechanism to alter the loop, re-injecting the next iteration (in *bach.roll* format), along with its possibly modified duration, in the module's lambda inlets. Due to the fact that such modifications are designed to operate continuously on the musical parameters, *cage.looper* only works with proportionally notated scores (*bach.roll*s). It should be remarked that, albeit originally based upon a timeshaping paradigm, the operation of *cage.looper* transcends the scope of this article, as it allows not only to modify the timing of the original score, but also its pitch content and, more generally, all its parameters in an iterative way.

3.4 Accelerandi and rallentandi

The *cage.agogics* abstraction allows to express rallentandi and accelerandi through a set of high-level musical parameters. The module combines the repetition functionalities of *cage.repeat* with the time distorsion and dilation approach provided by *cage.timewarp*, and indeed is based internally upon those modules.

The idea behind *cage.agogics* is to only expose control of three musically important parameters:

- N, i.e., number of repetitions of the source figure;
- D_E, i.e., the total duration of the accelerando/rallentando (i.e. the duration of the output score);
- r_{end}, i.e., the ending rate, determining how faster/slower the last repetition is, compared to the original (the output figure always starts with the same rate as the source score, i.e. $r_{\text{start}} = 1$).

The *cage.agogics* module assumes constant acceleration, which makes the three parameters not independent. In particular, let D_L be the total duration of the source score and let $r(t_L) = dt_L/dt_E$ be the remapping rate (and $r_{\text{end}} = r(ND_L)$), then the acceleration $a = \dfrac{r_{\text{end}} - 1}{ND_L}$ is assumed to be constant. Hence $a = \dfrac{r(t_L) - 1}{t_L}$ and by integration

$$t_E = \frac{\log(a \cdot t_L + 1)}{a}$$

and substituting $t_L = ND_L$ (as $t_E = D_E$) as well as the equation for a, one obtains the fundamental relation between the three musical parameters of *cage.agogics*:

$$\boxed{(r_{\text{end}} - 1)D_E = ND_L \log(r_{\text{end}})}$$

Only two of the three parameters are hence needed as input from the user—the third one is inferred.

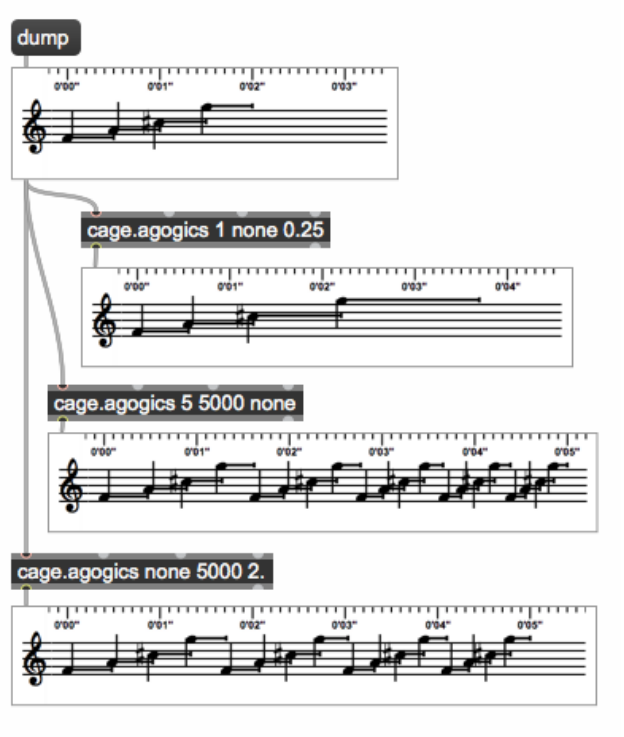

Figure 5. Three examples of usage of *cage.agogics*: the three arguments of the module are, respectively, the number of repetitions N, the total duration D_E and the ending rate r_{end}. The original cell is the uppermost score; then, from top to bottom: 1 repetition of the original cell ending 4 times slower; 5 repetitions of the cell lasting 5 seconds in total; a certain number of repetitions of the cell lasting 5 seconds and ending at about twice the original speed.

- If N and D_E are given, then

$$r_{\text{end}} = -\frac{ND_L}{D_E} W\left(-\frac{D_E}{ND_L} e^{-\frac{D_E}{ND_L}}\right),$$

  where W is the Lambert function (product logarithm);

- If N and r_{end} are given, then

$$D_E = ND_L \frac{\log(r_{\text{end}})}{r_{\text{end}} - 1};$$

- If D_E and r_{end} are given considered that N has to be integer, we take

$$N = \text{round}\left[\frac{(r_{\text{end}} - 1)D_E}{D_L \log(r_{\text{end}})}\right],$$

  and then proceed to recalculate r_{end} starting from D_E and the newly found N.

It could be worth remarking that the score subject to repetition does not necessarily have to be the actual material intended as the final musical result of the process. The repetition paradigm is a simple one to manage, hence its choice, but the process can be as well applied to an elementary cell, such as a single note, whose role is that of constituting a rhythmic grid which will serve as a lattice for a further compositional process. This of course is true for all the other operations described in this article, but seems particularly relevant with respect to this specific process, whose very formulation might otherwise appear arbitrarily restrictive.

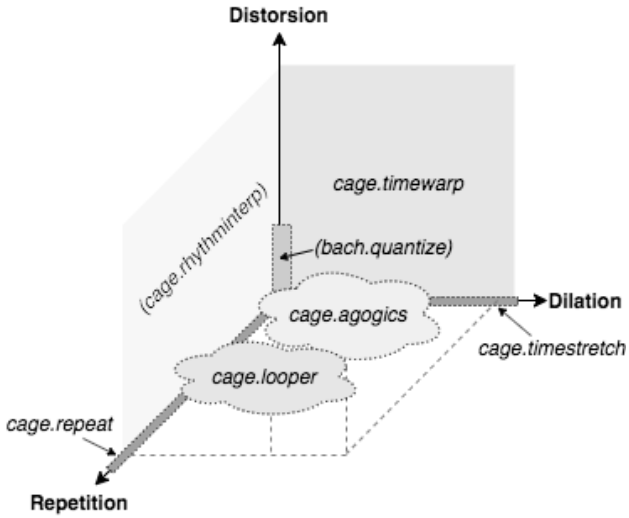

Figure 6. The organization of timeshaping modules of *cage* inside the space of Fig. 2.

3.5 Rhythmic interpolation

Rhythmic interpolation is achieved, in *cage*, by using the *cage.rhythminterp* abstraction. Differently from the previously described modules, this abstraction does not operate directly on scores, but rather only takes the rhythmical parameters into account (onsets and durations). This may however be interpreted as a way to achieve a certain kind of temporal distortion. The general idea is that, given a number n of rhythm-only scores, each containing m notes, it is possible to create a new one whose features lie somehow "in between" the original ones. Let us assume that o_j^i and d_j^i are respectively the onset and the duration of the j-th note of the i-th score ($i = 1, \ldots, n$, $j = 1, \ldots, m$); then we can build an interpolated score in a straightforward manner, having onsets $\overline{o}_j = \sum_i w_i o_j^i$ and durations $\overline{d}_j = \sum_i w_i d_j^i$, for $j = 1, \ldots, m$. The weights w_i determine the importance of each score in the interpolation and are normalized such that $\sum_i w_i = 1$.

The case where the original scores do not contain the same number of notes is more complex. The way we chose for tackling it actually treats all the scores as if they contained the same amount of notes, with some notes being duplicated so as to provide a fixed number of interpolation paths. For example, let us suppose that the first score contains three notes A_1, A_2 and A_3, and the second score contains two notes, B_1 and B_2. The idea behind the interpolation model adopted is that it is possible to build four lines upon which the interpolation happens: A_1 to B_1, A_3 to B_2, A_2 to B_1 and A_2 to B_2 (see Fig. 7). In principle the interpolation actually produces four notes, but when two (or more) of them have both their starting and ending times within a given threshold ϵ they are collapsed into a single one: in

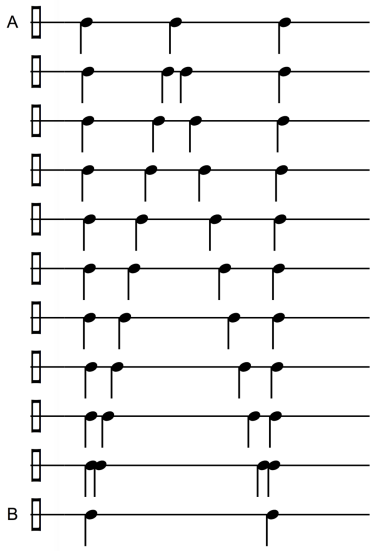

Figure 7. Rhythmic interpolation of two scores A and B containing a different number of notes (in 9 internal steps).

this way, at least the initial and final interpolation points will coincide with the original rhythmic structures.

4. QUANTIZATION

As hinted at before, we can consider rhythmic quantization as a very specific process of temporal distortion. Whether it should be included in this discussion is debatable, as quantization is generally considered a purely technical process, rather than a compositional tool for the musical elaboration of symbolic materials, since its aim is most often to convert a non-measured score into a measured one, or to simplify the rhythmic spelling of a pre-existing score. Nonetheless, in the practice of composers such as Emmanuel Nunes, quantization processes are arguably used with a somewhat direct aesthetic goal [17]: we shall therefore briefly discuss some aspects of quantization here; a more complete treatment will be perhaps the subject of a future article.

The term 'quantization' refers to the process of converting a score from proportional notation, where onsets and durations are real numbers representing some absolute timing, to standard notation, where onsets and durations are represented as rational subdivisions of a 'whole'. Quantization systems have a long history (see, in particular, [18, 19, 20]); in *bach*, quantization is achieved via the *bach.quantize* module, a rather complex one, with many different options and working mechanisms meant to address various specific cases which we shall not describe in detail here.

A simple quantization process could amount to 'snapping' the starting and ending positions of each note in a score to the nearest sixteenth subdivision. This operation would preserve the overall duration of a score, and alter its internal articulation through a consistent, deterministic rounding function of time against time: in this sense, it would precisely fit the definition of the distortion axis given above. Moreover, this operation would closely relate to a very idiomatic signal processing technique, bitcrushing, which is actually a kind of distortion performed through

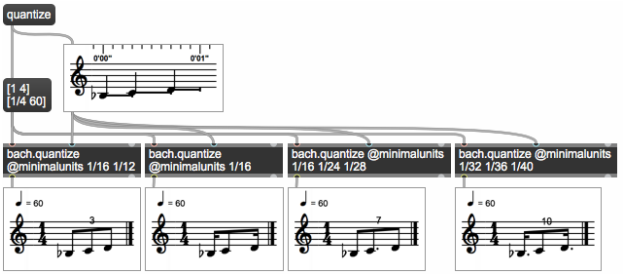

Figure 8. Temporal distortions given by quantizations of a proportionally notated fragment with the same meter/tempo and different 'snapping grids'.

applying a rounding function of instantaneous amplitude against instantaneous amplitude to an audio signal.

More complex kinds of quantization, also possible through *bach.quantize*, typically choose different rounding functions according to the specific contents and context of a temporal window of the score, thus performing an adaptive approximation rather than a fixed one. In the case of *bach.quantize*, the specific function adopted for each temporal window is not defined analytically, but constructed by a backtracking algorithm.

5. EXAMPLES

5.1 Risset rhythms

Following [21], we can implement in *bach* and *cage* a process producing eternal accelerandi or rallentandi build upon a given 'tenor'; the patch is shown in Fig. 9 (the 'tenor' is represented in the upper *bach.roll*, the resulting pattern is obtained in the lower *bach.roll*). The main building blocks are indeed *cage.repeat* and *cage.timewarp*, controlled via a very precise warping function, along with a volume windowing (via *cage.volume*) and a possible transposition of the pitches—the transposition stage can be bypassed, decoupling the treatment of agogics from the pitch shifting. Assuming that for a layer of index $v \geq 0$ the source score has been looped 2^v times, the equation connecting the source time and the elapsed time is

$$t_E = D_L \left(\log_2 \left(\frac{t_L}{D_L} + 2^v \right) - v \right),$$

where D_L is the duration of the source score. For more information on how this formula is obtained and on the mathematics involved, see [21]. [4] The patch heavily relies on the programming capabilities provided by the *bach.eval* module [22].

5.2 Vinyl-like speed up of scores

We can model the effect of a vinyl or tape recorder speeding up or slowing down (i.e. combining time warping and pitch shifting), and applying it to a proportionally notated score.

[4] In our implementation, the T and τ variables mentioned in the paper are set to have the same value $T = \tau = D_L$.

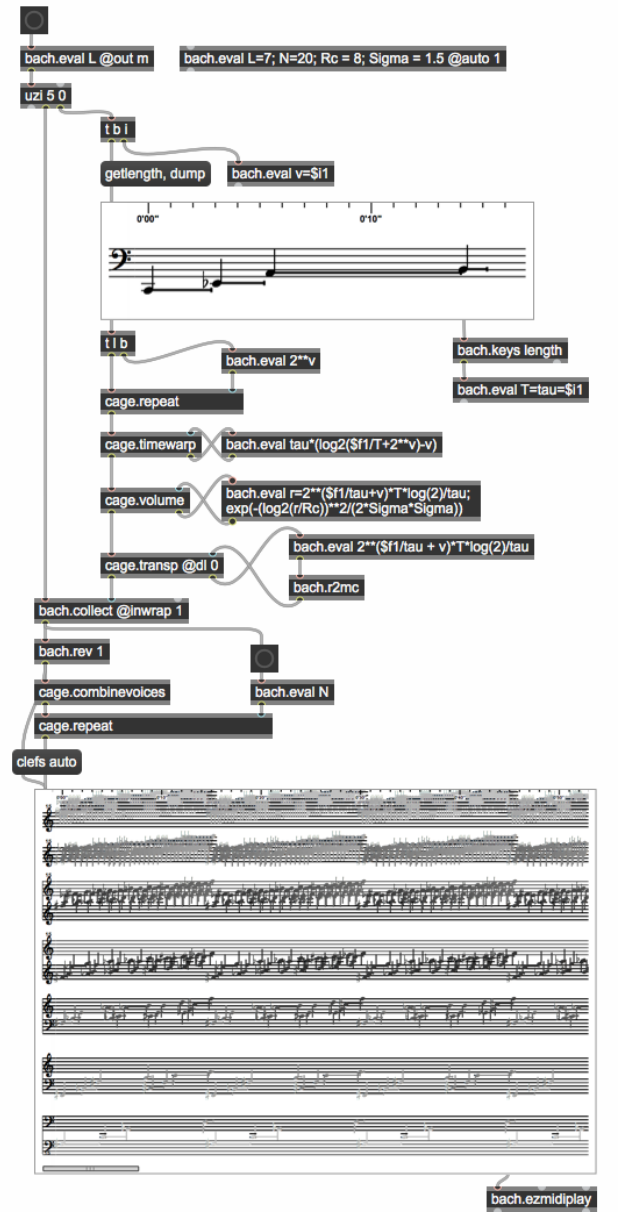

Figure 9. An example of patch producing Risset rhythms starting from a musical cell, implemented following [21] and used in Daniele Ghisi's 'Jean-Claude' (from the Rockenhausen Almanach)

Suppose you need to speed up the beginning of a score, until a certain note at onset $t_L = \overline{T}_S$. We can model the speeding up rate as $r(t_L) = (t_L/\overline{T}_S)^\alpha$ if $t_L < \overline{T}_S$, with $\alpha > 0$ being a reshaping exponent. Hence, for $t_L < \overline{T}_S$ one gets

$$t_E = \int_0^{t_L} \frac{1}{r(u)} du = \overline{T}_S^\alpha \int_0^{t_L} \frac{1}{u^\alpha} du = \frac{\overline{T}_S^\alpha}{1-\alpha} t_L^{1-\alpha}$$

with $0 < \alpha < 1$ needed for the integral to converge. This also yields a total speeding up time of $\overline{T}_S/(1-\alpha)$.

We remark that, conveniently, the slowing down portion can be seen as the retrogradation of the speeding up of the retrogradation.

Intuitively, this process yields a longer score than the original — unless one crops out only the final portion of the speeding up, and the initial portion of the slowing down in order to retain the absolute duration of each portion.

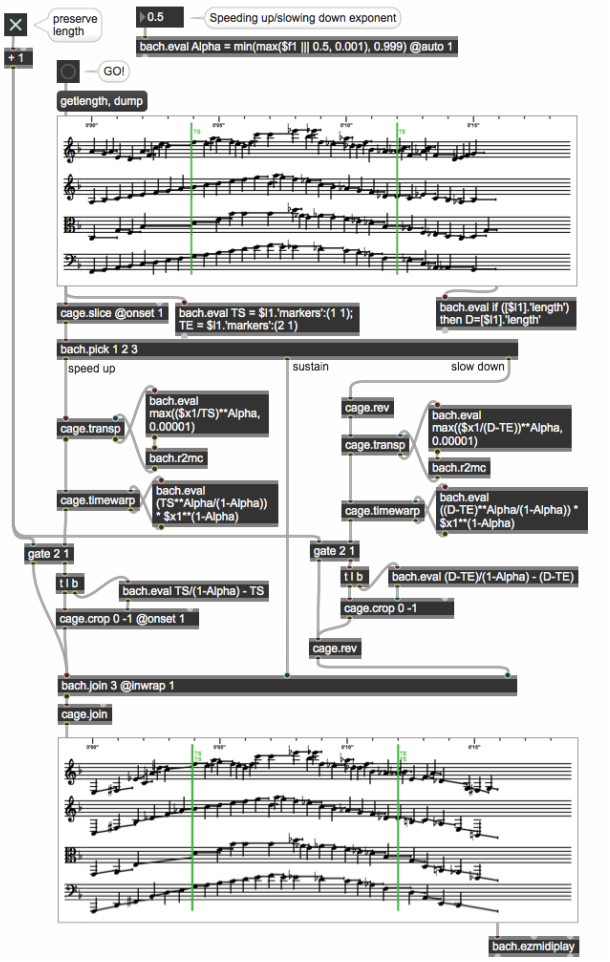

Figure 10. A patch producing a sped up (in the beginning) and slowed down (in the end) version of a source score (in this case, the beginning of Ravel's quartet). The shape of the agogics can be controlled via an exponent; the length of the original score can be preserved (via trimming). This process is used in Daniele Ghisi's music for 'La Chute'.

The patch implementing the process, including the possible cropping, is shown in Fig. 10.

6. CONCLUSIONS AND FUTURE WORK

We have organized the collection of 'timeshaping' practices, which amount to modifying the temporality of a score, into three different axes (dilations, distorsions and repetitions), and we have shown how some common musical practices lie within this categorization. We maintain that interactive computer-aided composition is an important, innovative approach to bridging the dual nature of time (linear, when it concerns becoming, and non-linear as source time). In particular, we have described how the *cage* timeshaping modules fit into the geometry of the aforementioned axes and we have provided some examples of applications.

There are still many open areas of interest. From a music theory perspective, the concept of interactive timeshaping needs to be explored further: we have hinted at some of the

implications in section 1.2, which may be, in the future, the starting point for a more detailed research on compositional practices. From a mathematical perspective, one may attempt to characterize the space defined in section 2 more formally; as an example, we think it would be interesting to investigate a general framework for temporal distorsions. These models may then have implications on the development of new computer aided-composition tools, such as a module for temporal distorsions, controlled via high-level musical parameters. This module might possibly stem from an extension of *cage.agogics* to account for non-uniform acceleration and variable starting rate. This would also in part overlap with the possibility of adding an 'order 2' attribute to *cage.timewarp*, allowing users to define acceleration in the lambda loop. Furthermore, quantization has a series of problem of its own, ranging from the capability to adapt the behavior to the musical processes, to the search for semi-automatic, dynamical meter and tempo inference tools.

7. REFERENCES

[1] J. P. Burkholder, "The uses of existing music: Musical borrowing as a field," *Notes*, vol. 50, no. 3, pp. 851–870, 1994. [Online]. Available: http://www.jstor.org/stable/898531

[2] J. Boyle and J. Jenkins, *Theft! A History of Music*. Duke Center for the Study of the Public Domain, 2017.

[3] D. Ghisi, "Music Across Music: Towards a Corpus-Based, Interactive Computer-Aided Composition," Ph.D. dissertation, Université Pierre et Marie Curie, Université Paris-Sorbonne, Sorbonne Université, IRCAM (UMR STMS 9912), 2017.

[4] K. Haddad, "Timesculpt in openmusic," in *Agon, C., Assayag, G. et Bresson, J., éditeurs: The OM Composer's Book*. Ircam/Delatour, 2006, vol. 1.

[5] C. Agon, "OpenMusic : Un langage visuel pour la composition musicale assistée par ordinateur." Ph.D. dissertation, University of Paris 6, 1998.

[6] M. Laurson and M. Kuuskankare, "PWGL: A Novel Visual Language based on Common Lisp, CLOS and OpenGL," in *Proceedings of International Computer Music Conference*, Gothenburg, Sweden, 2002, pp. 142–145.

[7] T. Baca, J. W. Oberholtzer, J. Trevino, and V. Adán, "Abjad: An open-source software system for formalized score control," in *Proceedings of The First International Conference on Technologies for Music Notation and Representation*, 2015.

[8] M. Puckette, "Max at seventeen," *Computer Music Journal*, vol. 26, no. 4, pp. 31–43, 2002.

[9] A. Agostini and D. Ghisi, "Real-time computer-aided composition with *bach*," *Contemporary Music Review*, no. 32 (1), pp. 41–48, 2013.

[10] A. Agostini, E. Daubresse, and D. Ghisi, "*cage*: a High-Level Library for Real-Time Computer-Aided Composition," in *Proceedings of the International Computer Music Conference*, Athens, Greece, 2014.

[11] P. Desain *et al.*, "Putting Max in Perspective," *Computer Music Journal*, vol. 17, no. 2, pp. 3–11, 1992.

[12] A. Agostini, D. Ghisi, and J.-L. Giavitto, "Programming in style with bach," in *Computer Music Multidisciplinary Research*, 2019.

[13] J. D. Kramer, *The Time of Music: New Meanings, New Temporalities, New Listening Strategies*. G. Schirmer, 1988.

[14] E. Maestri, "Notation as temporal instrument," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'16*, R. Hoadley, C. Nash, and D. Fober, Eds. Cambridge, UK: Anglia Ruskin University, 2016, pp. 226–229.

[15] H.-J. Rheinberger, "Transpositions: From Traces through Data to Models and Simulations," in *Transpositions: Aesthetico-Epistemic Operators in Artistic Research*, M. Schwab, Ed. Leuven University Press, 2018.

[16] A. Agostini and D. Ghisi, "A Max Library for Musical Notation and Computer-Aided Composition," *Computer Music Journal*, vol. 39, no. 2, pp. 11–27, 2015/10/03 2015. [Online]. Available: http://dx.doi.org/10.1162/COMJ_a_00296

[17] K. Haddad, "Fragments de recherche et d'expérimentation : Eléments de réflexions autour de l'écriture rythmique d'emmanuel nunes," https://medias.ircam.fr/x8f9800_karim-haddad-fragments-de-recherche-et-d, accessed: 2020-03-26.

[18] C. Agon, G. Assayag, J. Fineberg, and C. Rueda, "Kant: a critique of pure quantification." in *ICMC*, 1994.

[19] K. Sprotte, M. Laurson, and M. Kuuskankare, "Ksquant-complex score manipulation in pwgl through quantization," in *International Symposium on Computer Music Modeling and Retrieval*. Springer, 2008, pp. 253–261.

[20] A. Ycart, F. Jacquemard, J. Bresson, and S. Staworko, "A supervised approach for rhythm transcription based on tree series enumeration," in *Proceedings of the TENOR Conference*, Cambridge, UK, 2016.

[21] D. Stowell, "Scheduling and composing with risset eternal accelerando rhythms," in *Proceedings of the International Computer Music Conference*, 2011.

[22] A. Agostini and J. Giavitto, "*bell*, a textual language for the bach library," in *Proceedings of the International Computer Music Conference (to appear)*, New York, USA, 2019.